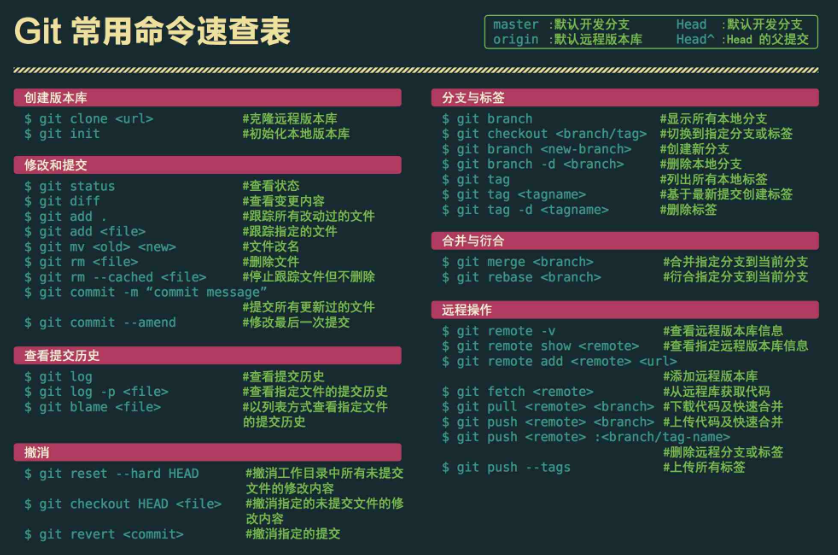
Git ：目前世界上最先进的分布式版本控制系统

git的最基本作用是版本控制，比如你写一个文档，老板一直让你改，这时笨的人会在原文档上直接改，聪明一点的会复制一下原文档，在副本上修改，这时老板说你的第三版比较好，笨的人煞笔了。。，聪明一点的就直接拷贝了第三版给客户发过去了。然后聪明一点的认为这次的事情完事了把其他没用的都删了，突然有一天老板说客户的意见和你最后一版的相同，这时聪明一点的人也煞笔了。。。。

来了个最聪明的人，会用git,他在本地建了一个版本库，每次老板让他修改，他就把之前的版本提交一下，并标明这版的主要特点，这样文件夹里就只有一个文档，每次老板说要那个版本的，他就直接从版本库里恢复一下。

git通常在编程中会用到，通常是多个人同时协作一个项目，有可能出现两个人同时修改一个文件，这时后提交的人会遇到冲突，需要解决冲突；git能够记录每个人的提交修改等形成日志，可以根据提交记录进行回滚；并且git支持分布式部署。



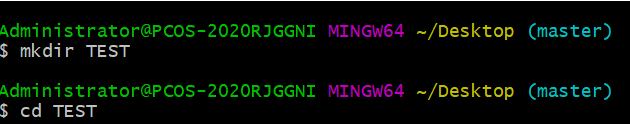
流程：1. 远程（remote）通过克隆或者提取的方式到仓库区（repository）再通

过查看的方式到工作区（workspace）

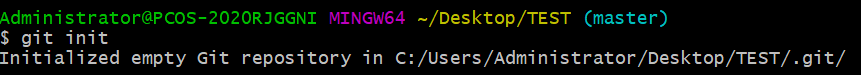
1. 远程（remote）直接通过pull拉取的方式到工作区（workspace）
2. 工作区（workspace）通过添加的方式到暂存区（index），再通过付（commit)的方式到仓库区（repository），仓库区（repository）再通过推送（push）的方式到远程（remote）

如何从本地推送文件到远端：

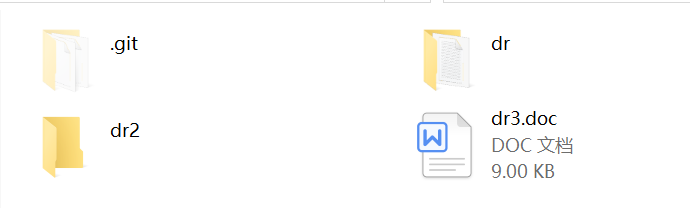
1.需要创建一个文件夹，可以右键直接新建，也可以通过git bash或者 分别输入以下操作 例如建立TEST



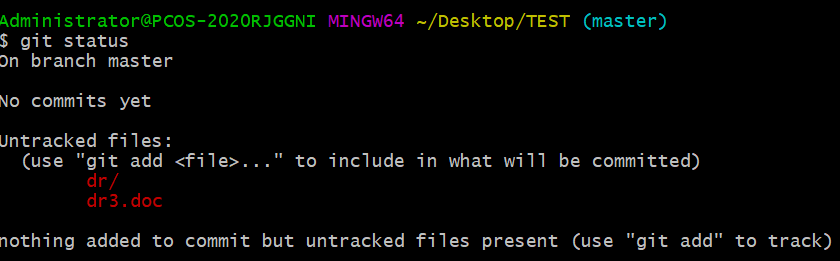
1. 通过命令git init 把文件夹变成git可管理的仓库



1. 在TEST里添加文件，不要.git里添加



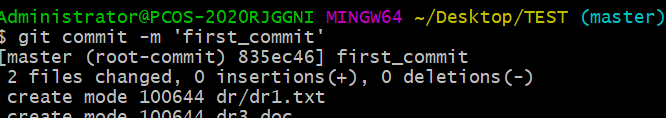
1. 用git status查看状态



1. 通过git .add .把项目添加到仓库上

IMG_256

6.用git commit -m‘first commit’ 提交到仓库

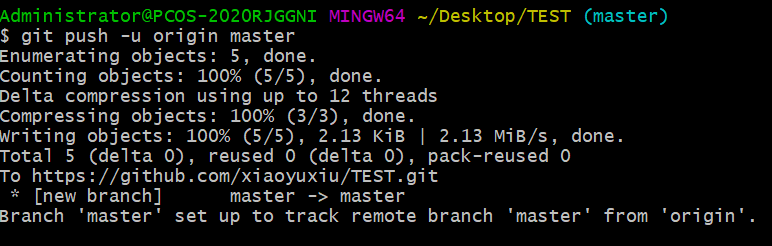


1. 通过 git remote add origin+远程库地址来关联远程库

IMG_256

8.若远程库不为空，则必须通过git pull --rebase origin master获取远程库与本地库合并

9.用git push上传项目



PS；git pull --rebase 表示把你的本地当前分支里的每个提交(commit)取消掉并且把它们临时 保存为补丁(patch)(这些补丁放到".git/rebase"目录中),然后把本地当前分支更新 为最新的"origin"分支，最后把保存的这些补丁应用到本地当前分支上。

读取他人项目操作：

1. 在搜索框搜索他人仓库（对方名字/仓库名）
2. 点击进入
3. 点击fork，就可以把他人仓库拷贝到本地

1604672823(1)

1. 用笔可对文件进行修改，直接在空白处修改，则你的本地项目有变化，但他人的项目还是没有变化

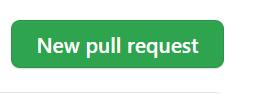
1604672964(1)

如何让你改变项目之后也保存在他人的项目中：

1. 点击pull request

1604673228(1)

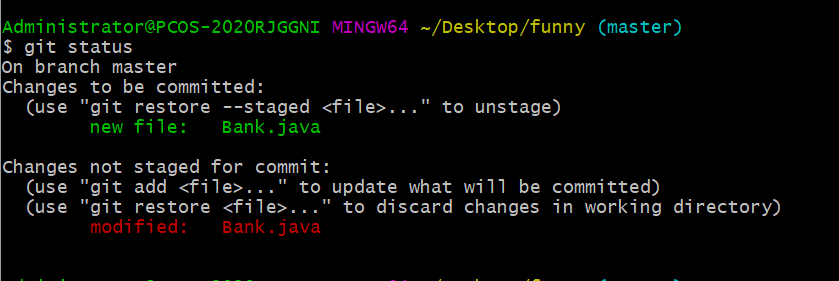
1. 再点击New pull requests



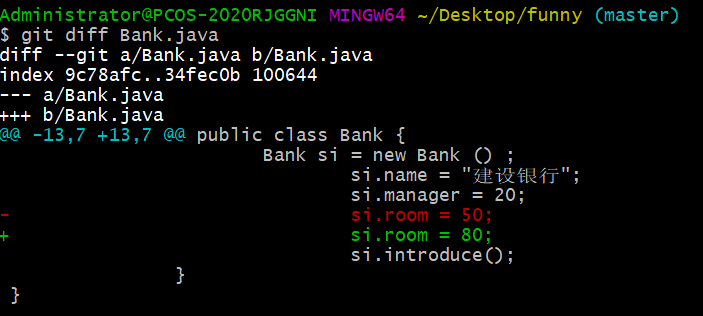
1. 系统会自动把两个文件进行比对，再点击create pull request ，空白处为说明信息，可写可不写，再点击create pull request，要是他人同意了，则他人可以合并起来.他人收到消息后，可以在pull request点开后直接进去查看修改的内容，若是要合并，则点击Merge pull request，再点击confirm merge

检查修改的文件：（文件放进去都要提交）

1. 我们在文件夹里把要提交的文件修改之后，bash会显示我们修改了文件，

但不显示修改的内容（我们将源文件的20改成80）

1. 用git diff+文件名 查看内容

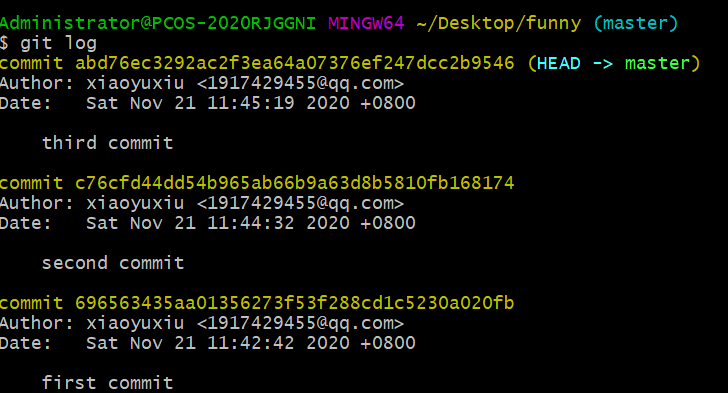


1. 知道改了哪里之后若想提交，先add，再用git status再查看一下状态 ，确定无误再commit到仓库里



版本回退：

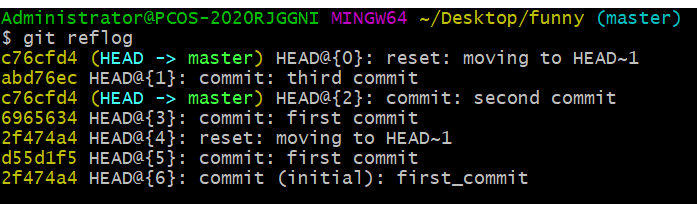
1.修改文件后，继续对文件进行修改，修改了三次，要想查看记录， 使用 git log查看 显示三次

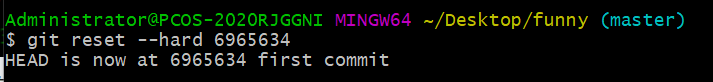


2.把当前版本回退到上一版本，用git reset --hard HEAD~数字 ，例如回退到上一版本则为git reset --hard HEAD~1，先查看内容则用 cat+文件名 查看

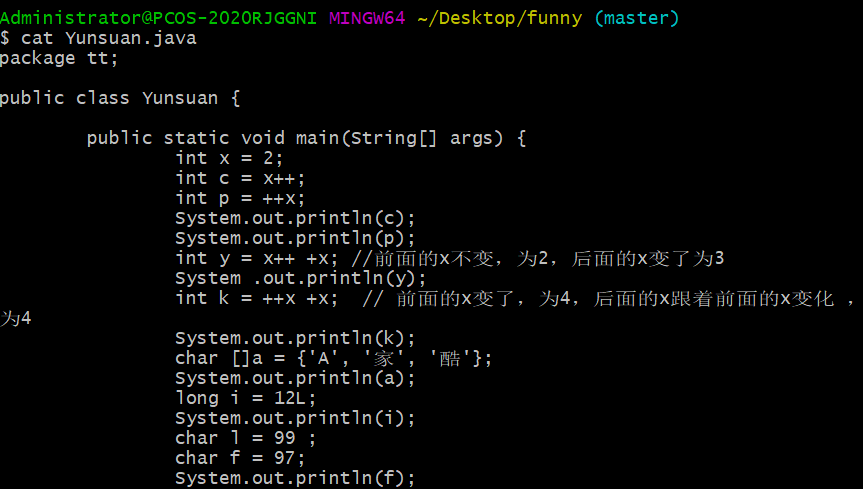


3.又想回退到最新版本，通过git reflog 找到版本号，再通过使用git reset --hard 版本号  找回





4.可通过cat+文件名查看是否已恢复



删除本地仓库：

1.进入项目所在目录

IMG_256

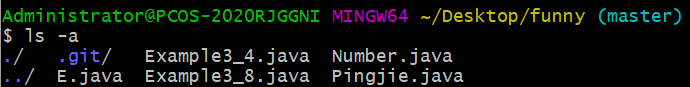
1. 显示所有本地分支（初始化只有一个master分支）

IMG_256

1. 初始化本地版仓库（重新初始化一次，可以忽略）

IMG_256

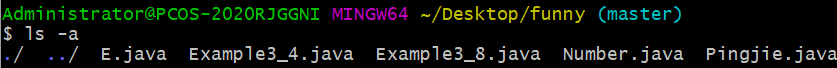
1. 找到目录下隐藏的 .git



1. 删除 .git

IMG_256

1. 查看是否删除



工作区和暂存区的区别：

工作区：就是你在电脑上看到的目录，比如目录下funny里的文件(.git隐藏目录版本库除外)。或者以后需要再新建的目录文件等等都属于工作区范畴。

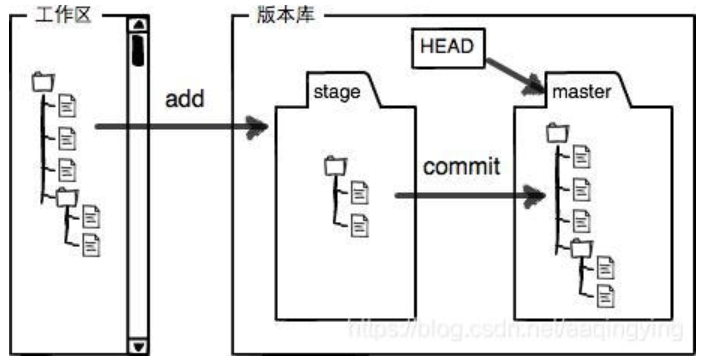
版本库(Repository)/仓库：工作区有一个隐藏目录.git,这个不属于工作区，这

版本库。其中版本库里面存了很多东西，其中最重要的就是stage(暂存区)，还有Git为我们自动创建了第一个分支master,以及指向master的一个指针HEAD。

Git提交文件到版本库有两步：

第一步：是使用 git add 把文件添加进去，实际上就是把文件添加到暂存区。

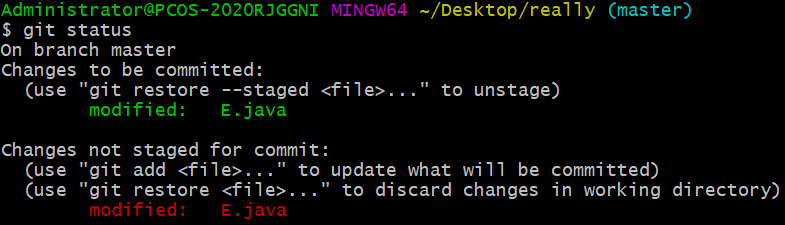
第二步：使用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支上。



Git撤销修改：

1 .发现添加的有误，想删除自己添加的东西（法一自己手动修改文件，法二用版本回退）

1. 在做撤销之前，可用git status先查看状态



3.利用git checkout -- 文件名撤销

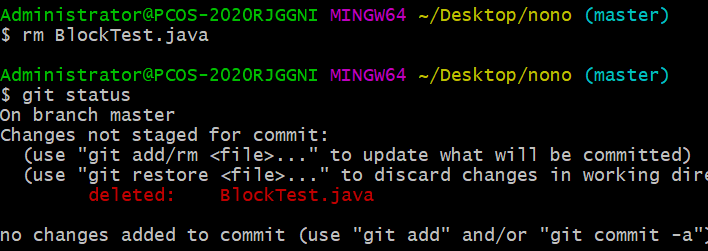
IMG_256

4.利用cat +文件名查看内容是否已撤销

5.注意：文件添加后没法撤销

Git删除文件：

1. 可直接在目录下删除文件或通过rm +文件名删除



1. 在没有提交之前，可用git checkout -- 文件名恢复

IMG_256

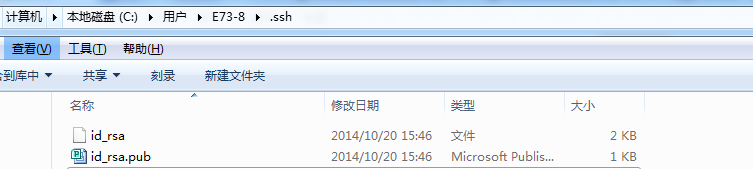
PS: git status命令用于显示工作目录和暂存区的状态。使用此命令能看到那些修改被暂存到了, 哪些没有, 哪些文件没有被Git tracked到。git status不显示已经commit到项目历史中去的信息。看项目历史的信息要使用git log.

创建远程仓库

在了解之前，先注册github账号，由于你的本地Git仓库和github仓库之间的传输是通过SSH加密的，所以需要一点设置：

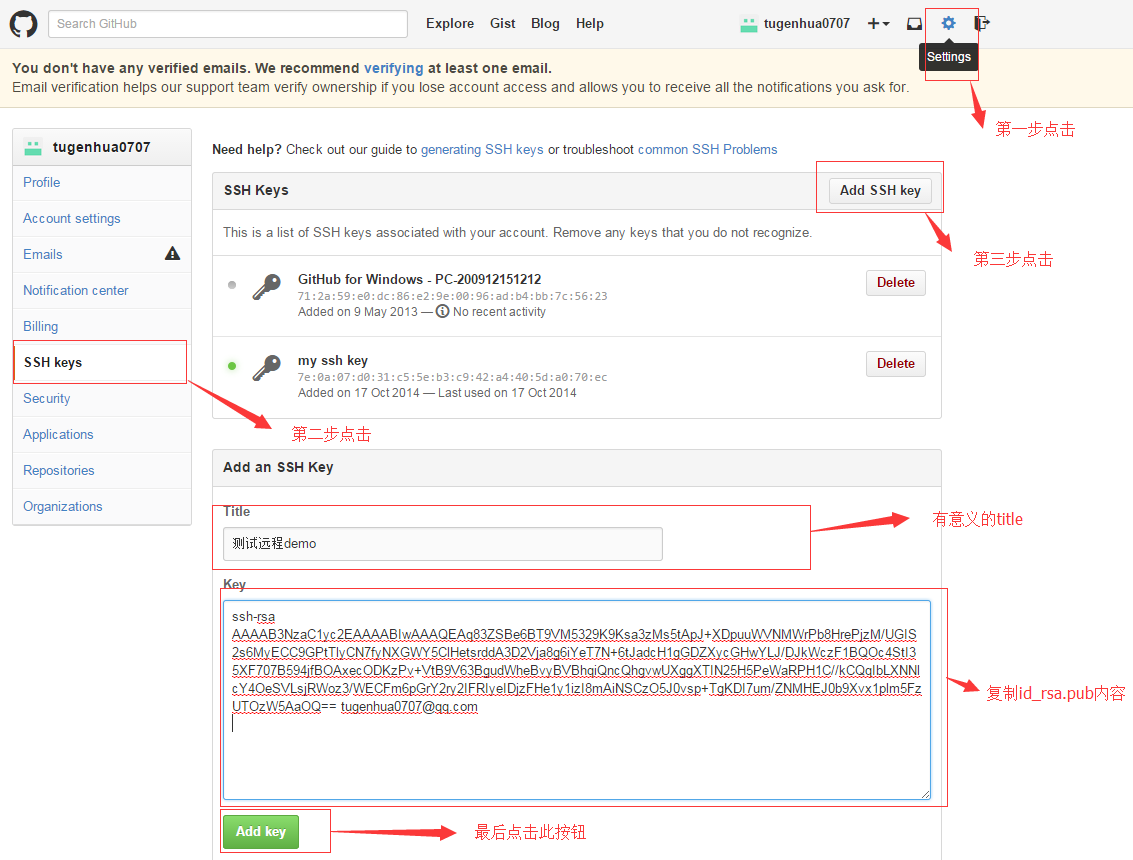
第一步：创建SSH Key。在用户主目录下，看看有没有.ssh目录，如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文件，如果有的话，直接跳过此如下命令，如果没有的话，打开命令行，输入如下命令：

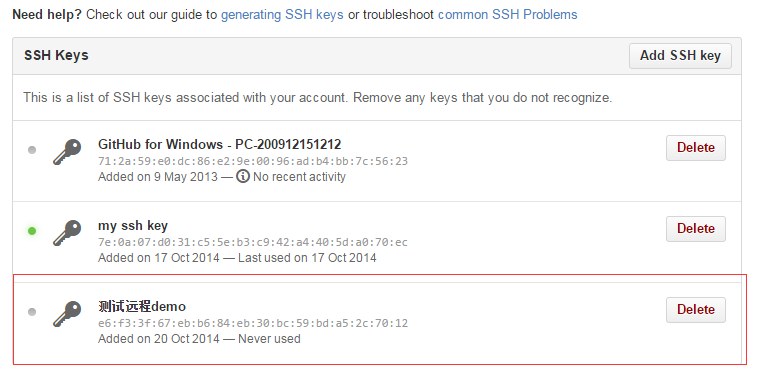
ssh-keygen -t rsa –C “youremail@example.com”, 由于我本地此前运行过一次，所以本地有，如下所示：



id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

第二步：登录github,打开” settings”中的SSH Keys页面，然后点击“Add SSH Key”,填上任意title，在Key文本框里黏贴id\_rsa.pub文件的内容。



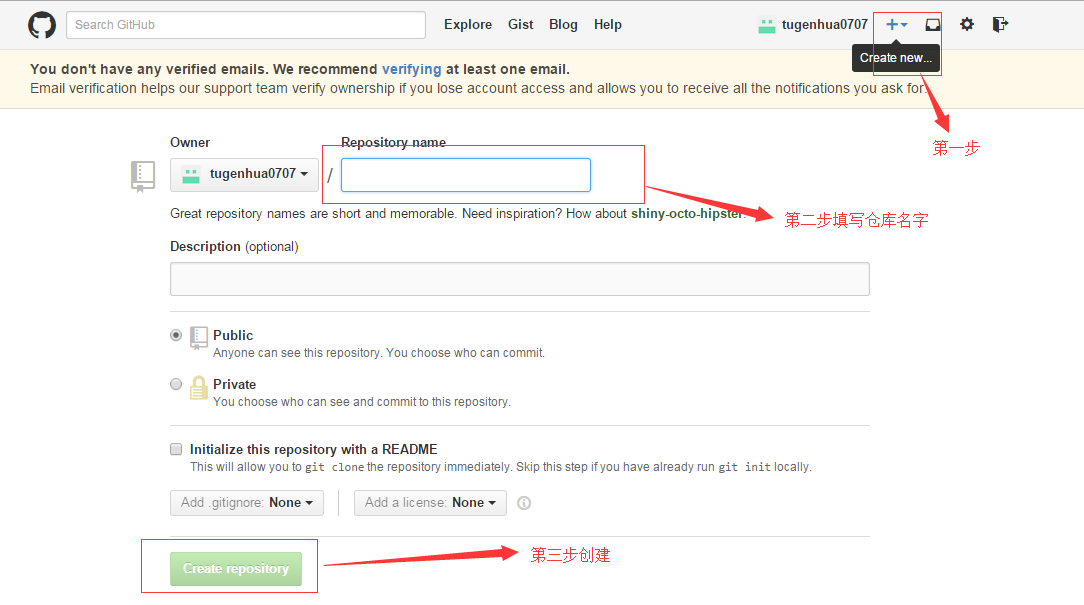
点击 Add Key，你就应该可以看到已经添加的key。  


如何添加远程库？  
现在的情景是：我们已经在本地创建了一个Git仓库后，又想在github创

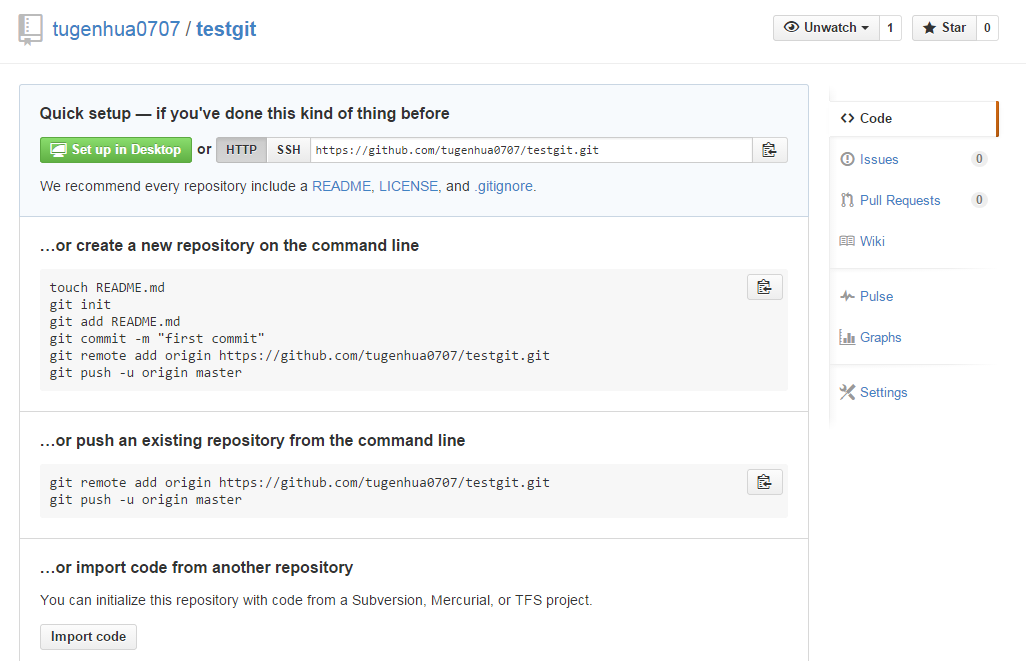
一个Git仓库，并且希望这两个仓库进行远程同步，这样github的仓库可以作为

备份，又可以其他人通过该仓库来协作。

首先，登录github上，然后在右上角找到“create a new repo”创建一个新的仓库。如下：



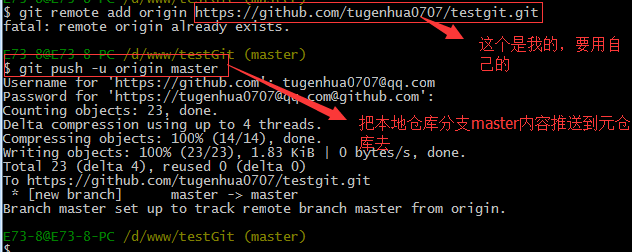
在Repository name填入testgit，其他保持默认设置，点击“Create repository”按钮，就成功地创建了一个新的Git仓库：



现在，我们根据GitHub的提示，在本地的testgit仓库下运行命令：

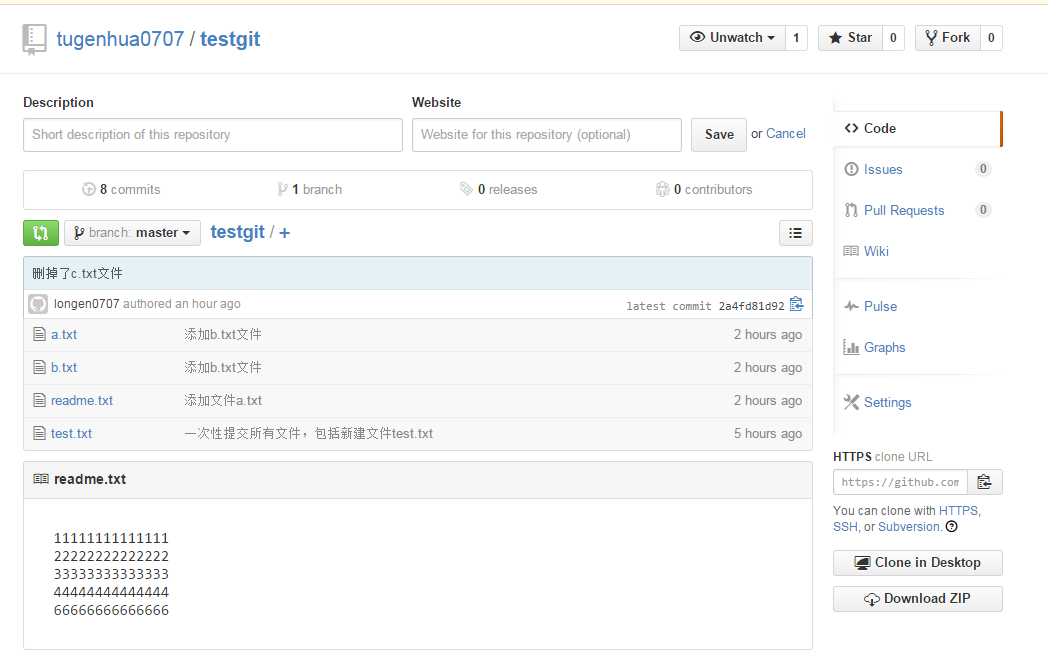
git remote add origin https://github.com/tugenhua0707/testgit.git

所有的如下：



把本地库的内容推送到远程，使用 git push命令，实际上是把当前分支master推送到远程。

由于远程库是空的，我们第一次推送master分支时，加上了 –u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。推送成功后，可以立刻在github页面中看到远程库的内容已经和本地一模一样了，上面的要输入github的用户名和密码如下所示：



从现在起，只要本地作了提交，就可以通过如下命令：

git push origin master

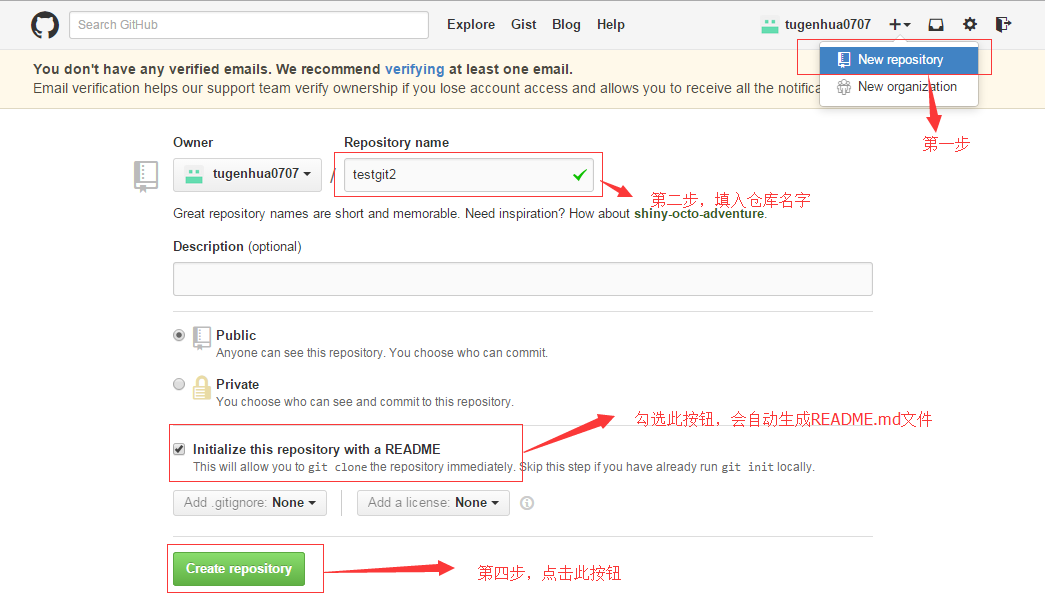
把本地master分支的最新修改推送到github上了，现在你就拥有了真正的分布式版本库了。

如何从远程库克隆？

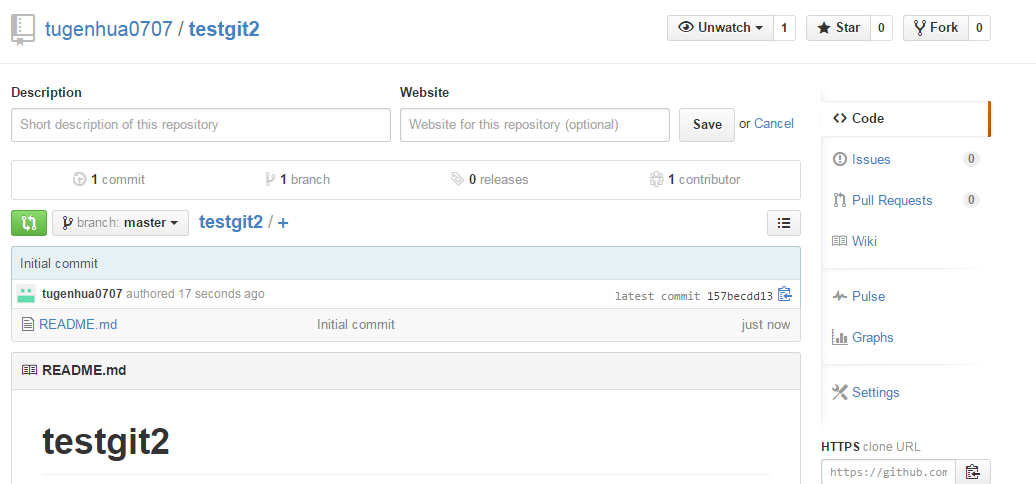
上面我们了解了先有本地库，后有远程库时候，如何关联远程库。

现在我们想，假如远程库有新的内容了，我想克隆到本地来 如何克隆呢？

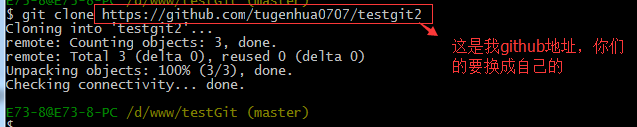
首先，登录github，创建一个新的仓库，名字叫testgit2.如下：



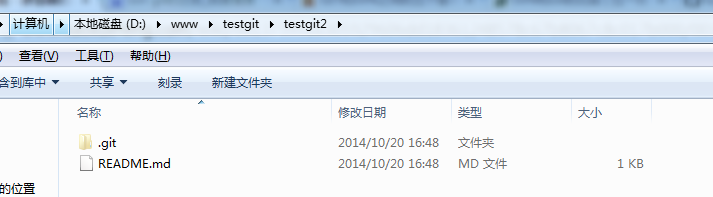
如下，我们看到：



现在，远程库已经准备好了，下一步是使用命令git clone克隆一个本地库了。如下所示：



接着在我本地目录下 生成testgit2目录了，如下所示：



创建与合并分支

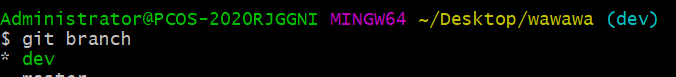
在 版本回填退里，你已经知道，每次提交，Git都把它们串成一条

间线，这条时间线就是一个分支。截止到目前，只有一条时间线，在Git里，这个分支叫主分支，即master分支。HEAD严格来说不是指向提交，而是指向master，master才是指向提交的，所以，HEAD指向的就是当前分支。

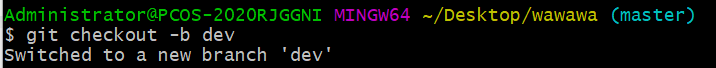
1. 创建分支 git branch+文件名

IMG_256

1. 切换分支 git checkout+文件名

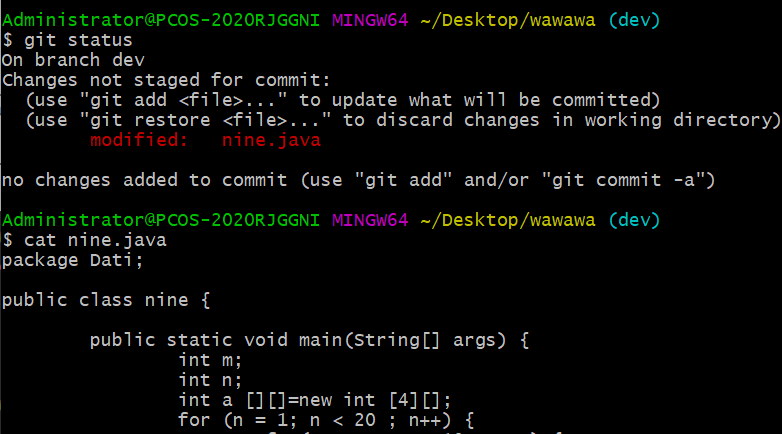


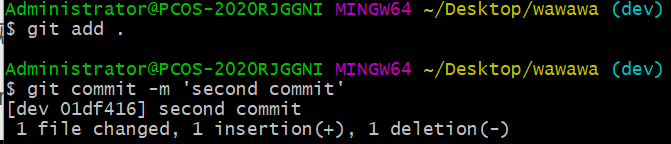
1. 也可直接创建+切换分支 git checkout -b +文件名



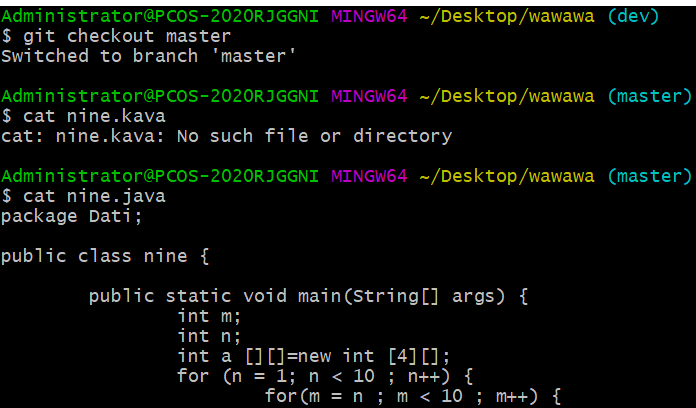
1. 在dev的分支做测试

先在dev分支上进行修改（改数字20）



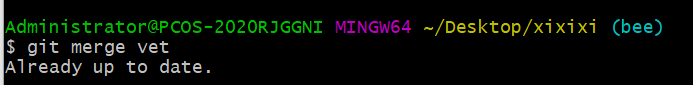


切换分支，再查看内容

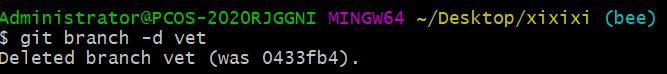


Master分支上没有显示修改后的内容，因为对其的修改是在dev分支上

1. 我们可以把dev分支合并到master分支 就可以在master分支上看修

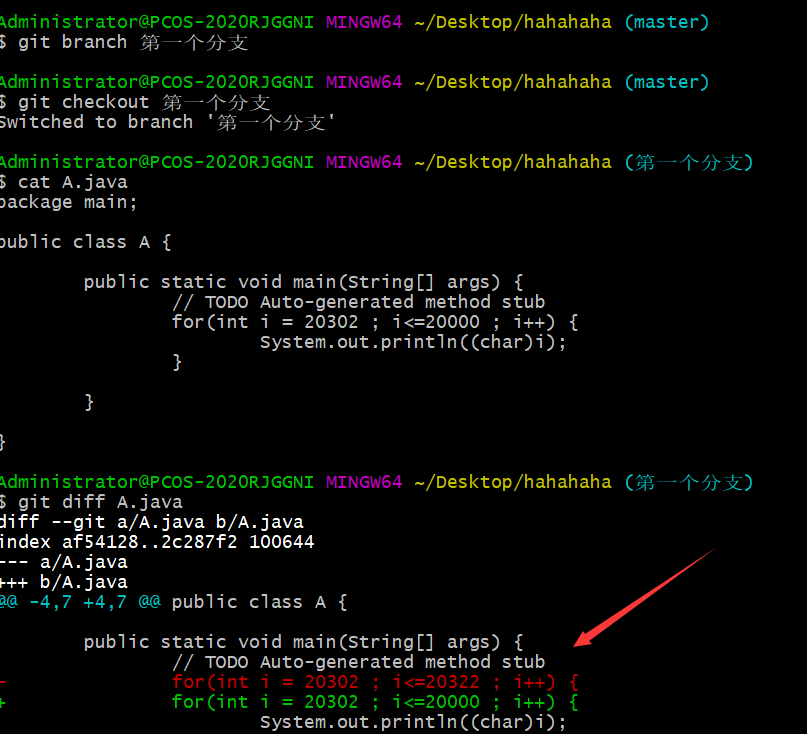
改内容 git merge （当前分支以外分支的名字） 

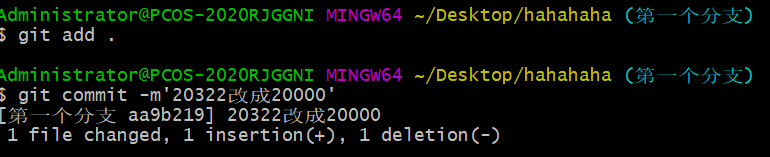
1. 删除分支 git branch -d + 文件名



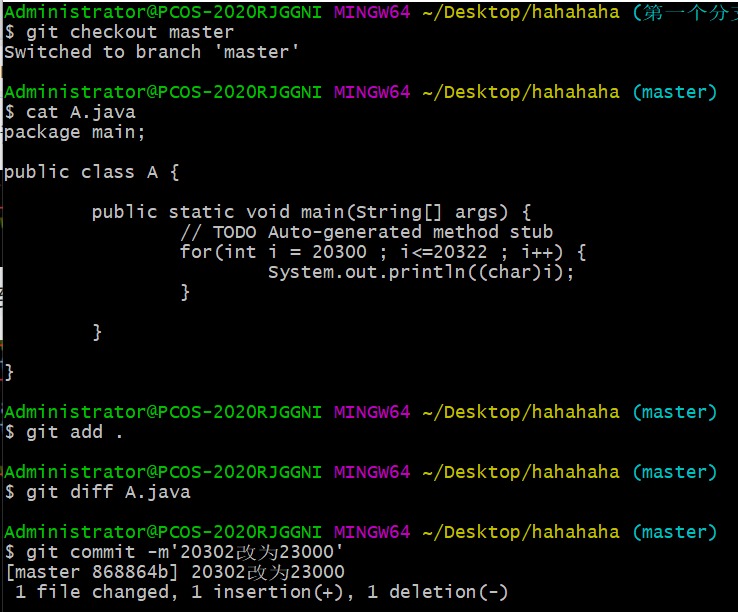
解决冲突

1. 新建第二个分支

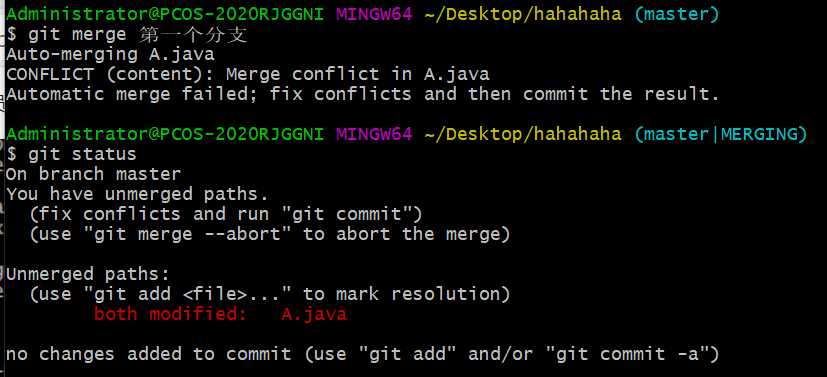




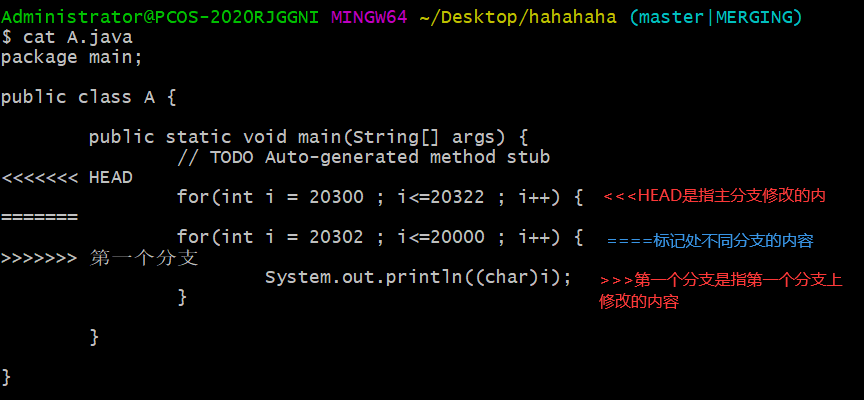
2.切换master分支修改内容



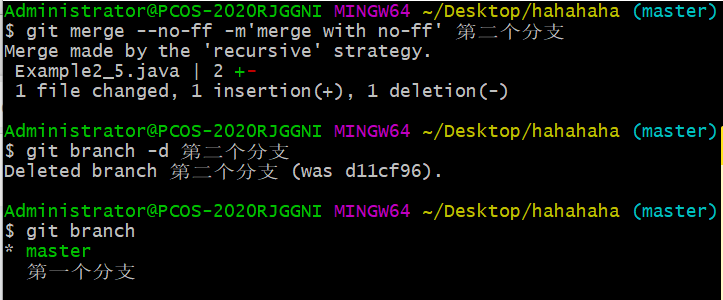
3.在master上合并第一个分支

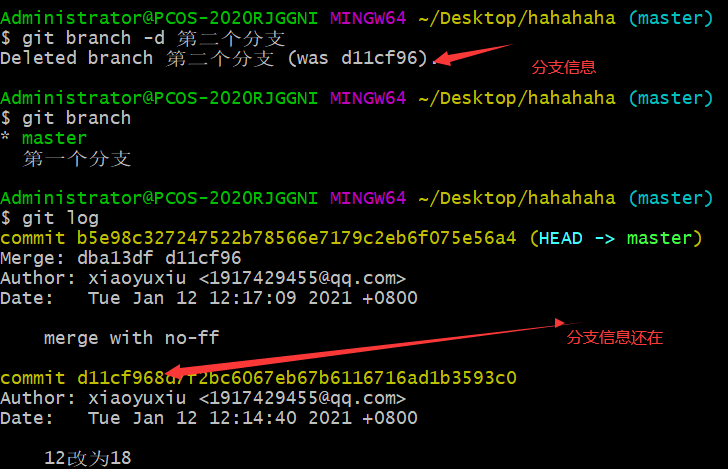


4.再查看内容，会出现冲突代码

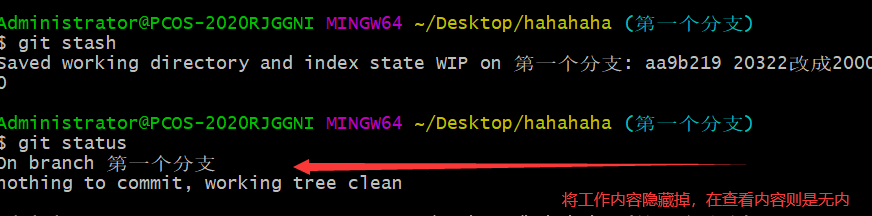


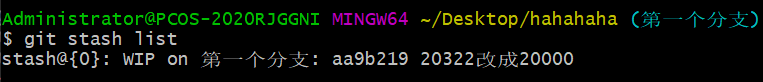
通常合并分支时，git一般使用”Fast forward”模式，在这种模式下，删除分支后，会丢掉分支信息，现在我们来使用带参数 –no-ff来禁用”Fast forward”模式



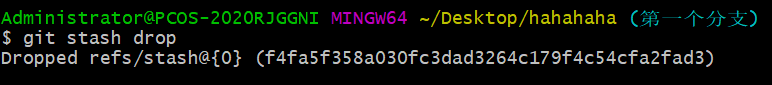


bug分支

1. 在当前的分支未提交时，却要先修改bug，只能先创建一个分支来修复
2. 这种情况可以先用stash功能“隐藏”当前分支
3. 接着在创建新分支修复bug，最后转化为master，再将两者合并，最后删除新分支
4. 再回到之前需要工作的分支，利用 git stash list恢复



1. 想删除则用 git stash drop



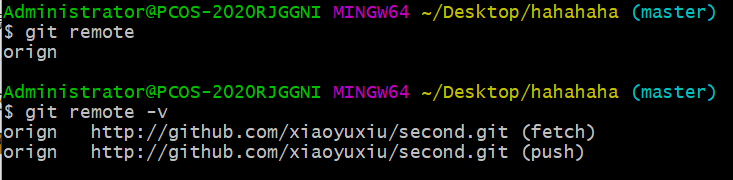
删除提交记录

1. git reset --hard HEAD~1
2. git push --force

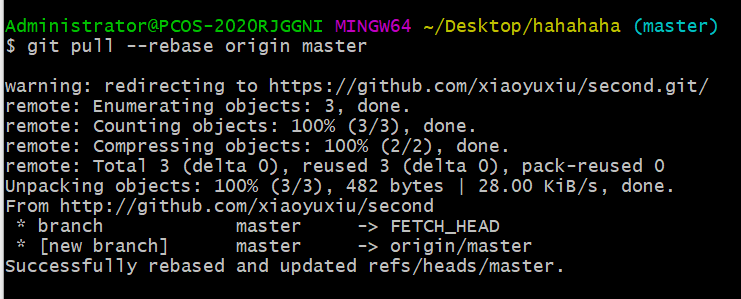
多人协作

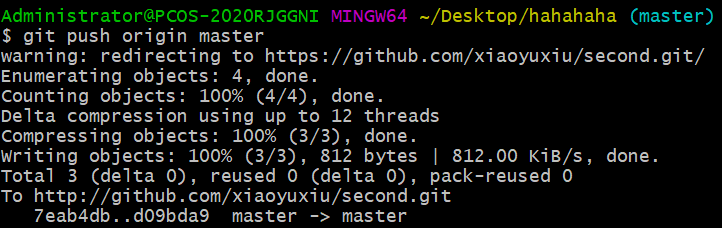
当从远程库克隆时，实际上git自动把本地的master分支和远程master、分支对应起来，远程库默认库名为origin

1. 关联远程库 git remote add origin +地址
2. 查看远程库信息 git remote
3. 查看远程库详细信息 git remote -v



注意：若远程库不为空，则必须通过git pull --rebase orign master获取远程库与本地库合并





推送分支

git push origin master （推送到主分支）

git push origin +分支名 (推送到分支上）

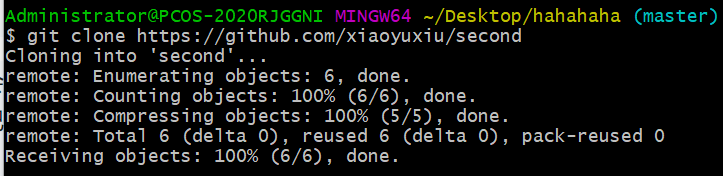
抓取分支

多人协作时，大家都会往master分支上推送各自的修改。

1. 把分支dev推送到远程去



1. 克隆远程库到本地 git clone + 地址

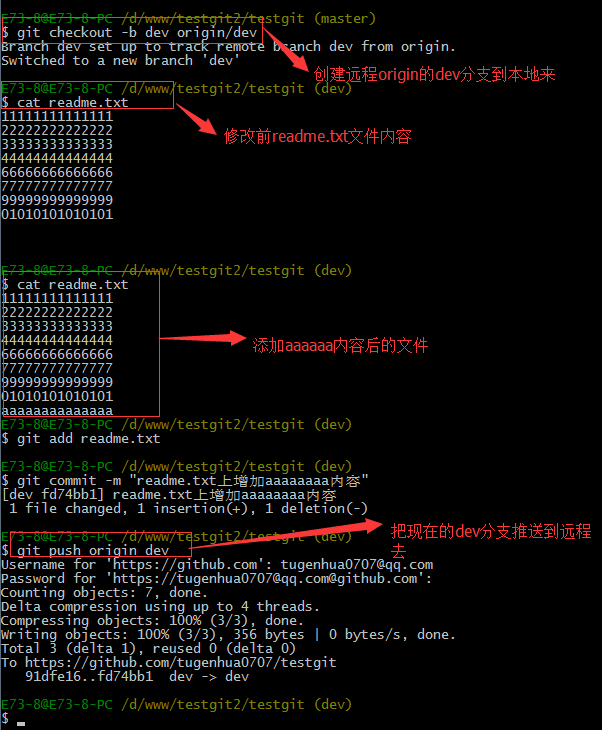


1. 目录生成



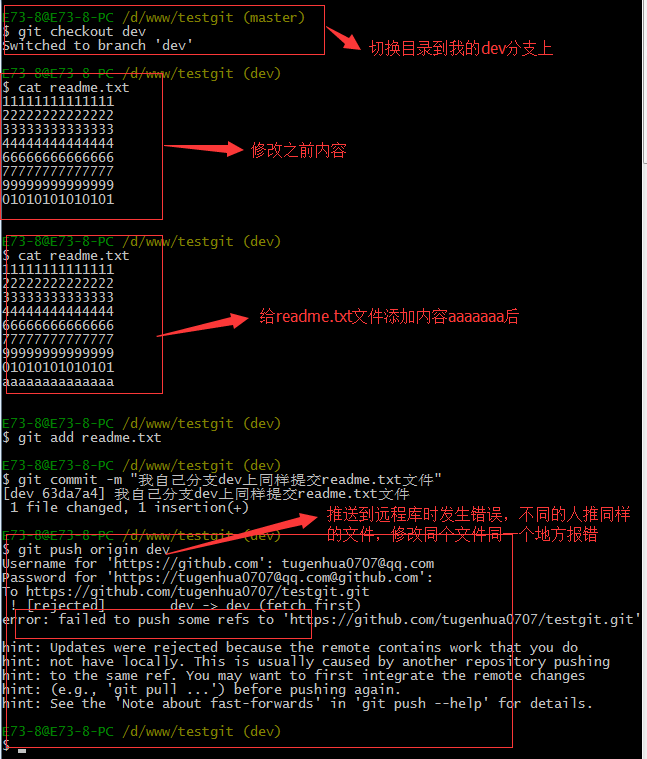
现在我们的小伙伴要在dev分支上做开发，就必须把远程的origin的dev分支到本地来，

4.使用命令创建本地dev分支：git checkout –b dev origin/dev，现在小伙伴们就可以在dev分支上做开发了，开发完成后把dev分支推送到远程库时。



小伙伴们已经向origin/dev分支上推送了提交，而我在我的目录文件下也对同样的文件同个地方作了修改

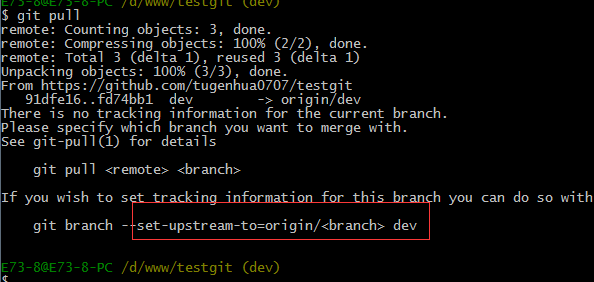
5.推送到远程



显示推送失败，因为我的小伙伴最新提交的和我试图推送的有冲突

解决方案

1. 先用git pull把最新的提交从origin/dev抓下来



1. git pull也失败了，则指定本地dev分支与远程origin/dev分支的链接



手动解决在提交

1. 再用 git push origin + 名称 进行推送

# **git 中出现>这个符号**

**Ctrl+D**